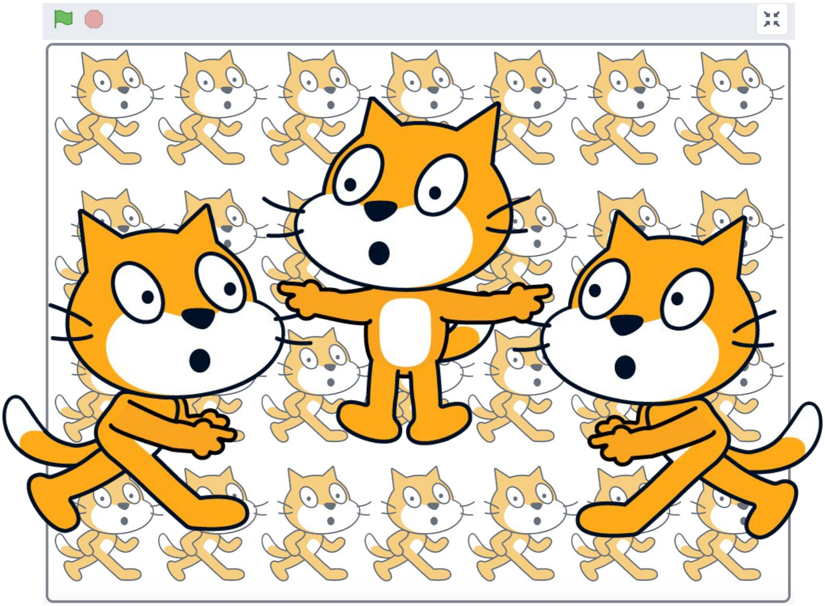
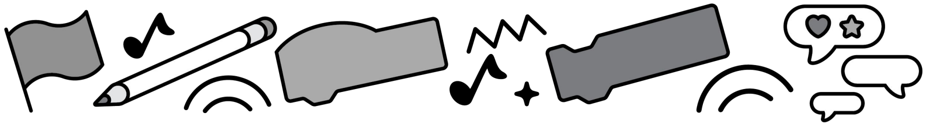
Advanced Topics: Clones



Use clones to create multiple sprites for more efficient and advanced projects.

scratch.mit.edu

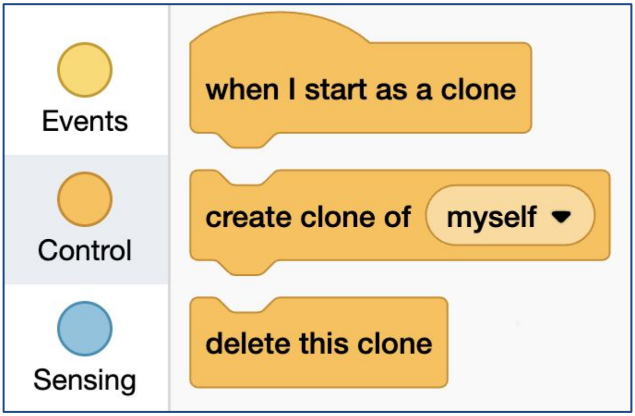
Set of 7 cards



Cards in This Pack

- Clone 101
- More Clone 101
- Balloon Pop Clones: Quick Game Creation
- Clone Piano: Using Local Variables
- Clone Piano: Using Global Variables
- Fractal Tree: Clones Making Clones
- Clone IDs: Generating and Using

Clone 101



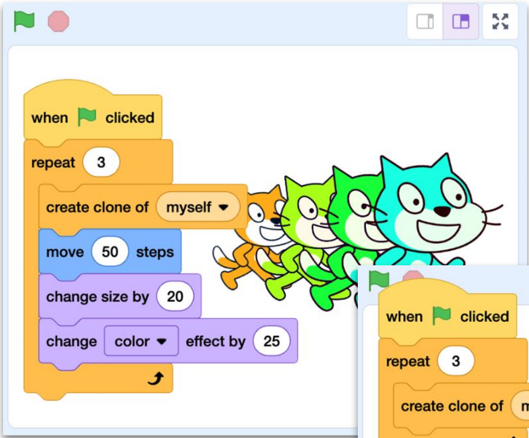
Cloning lets you create multiple copies of your sprite while your project is running. When each clone is produced, it has the **same costumes, sounds, scripts, and variables as the original, but it is otherwise independent**.

There are **three blocks that are specific to clones, which can be found in the Control category**: “create clone,” “delete this clone,” and “when I start as a clone” (where you can define code that only applies to the clones and not the original sprite).

Clone 101

scratch.mit.edu

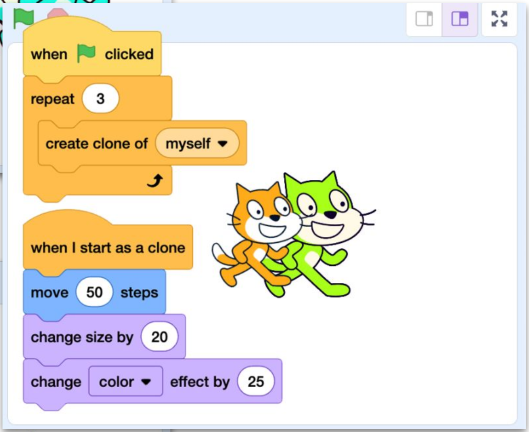
What differences do you notice in the code and the results between these three similar scripts?



```
when [flag] clicked
repeat (3)
    create clone of (myself ▾)
    move (50) steps
    change size by (20)
    change [color ▾] effect by (25)
```

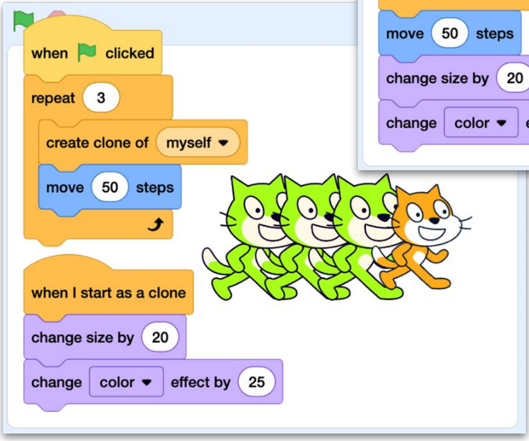
Sequence and project goals are important.

In the first case, the original sprite moves and changes, creating clones along the way.



```
when [flag] clicked
repeat (3)
    create clone of (myself ▾)

when I start as a clone
move (50) steps
change size by (20)
change [color ▾] effect by (25)
```

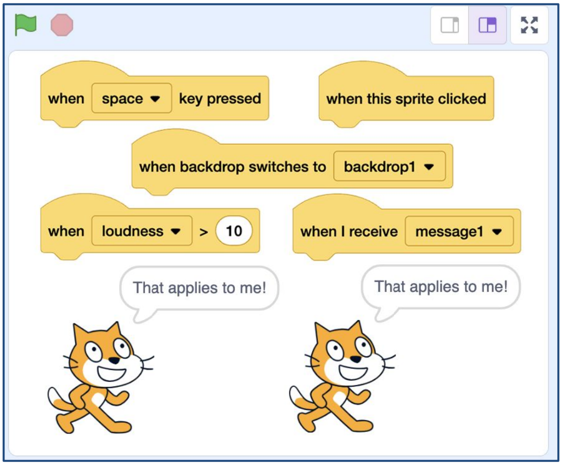


```
when [flag] clicked
repeat (3)
    create clone of (myself ▾)
    move (50) steps

when I start as a clone
change size by (20)
change [color ▾] effect by (25)
```

In other cases, the clone is created and then controls its own movement and/or effects.

More Clone 101



When a clone is created, the **scripts for the original sprite also apply to a clone**. That means that scripts like “when this sprite clicked” or “when I receive [broadcast]” apply to them, too.

One exception is scripts that have started running before the clone was created (like a “when green flag clicked” script). Code that should be applied to a clone needs to be triggered after their creation (via a broadcast, click, key press, “when I start as a clone,” etc.).

More Clone 101

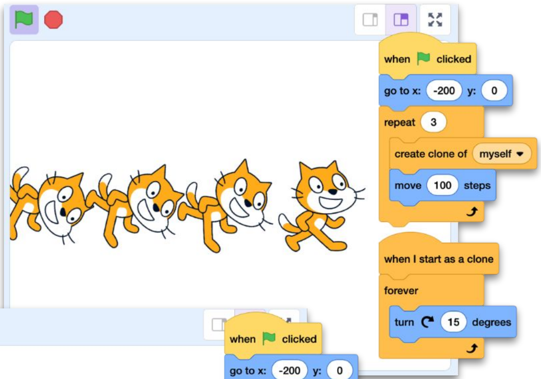
scratch.mit.edu

What differences do you notice in the code and the results between these three similar scripts?

How does timing and sequence affect when the clones start turning?

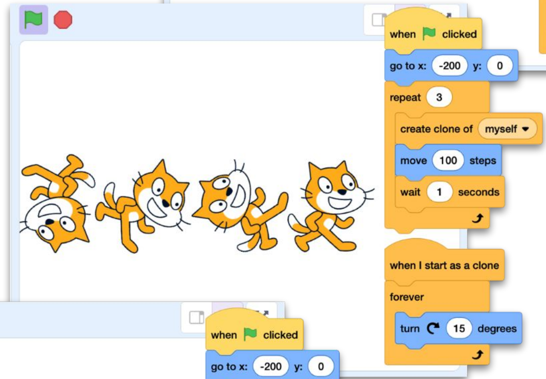
Why is the original sprite turning in one example and not the others?

What if the spin was triggered by clicking the sprite instead?



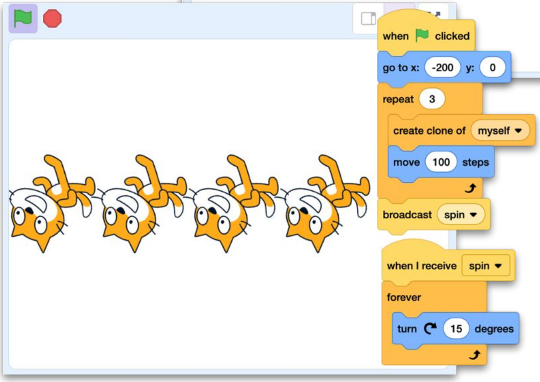
```
when [flag] clicked
go to x: -200 y: 0
repeat 3
    create clone of myself
    move 100 steps

when I start as a clone
forever
    turn ↻ 15 degrees
```



```
when [flag] clicked
go to x: -200 y: 0
repeat 3
    create clone of myself
    move 100 steps
    wait 1 seconds

when I start as a clone
forever
    turn ↻ 15 degrees
```

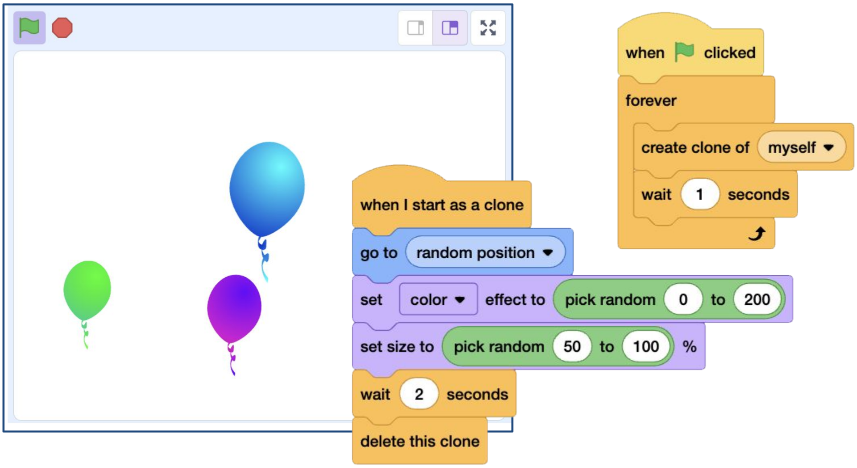


```
when [flag] clicked
go to x: -200 y: 0
repeat 3
    create clone of myself
    move 100 steps
broadcast spin

when I receive spin
forever
    turn ↻ 15 degrees
```

What is the difference between using a broadcast to start the turning vs attaching the forever sequence after the repeat sequence?

Balloon Pop Clones: Quick Game Creation



```
when ⚑ clicked
forever
    create clone of (myself ▾)
    wait (1) seconds
↺
```

```
when I start as a clone
go to (random position ▾)
set [color ▾] effect to (pick random (0) to (200))
set size to (pick random (50) to (100)) %
wait (2) seconds
delete this clone
```

You can **use clones to create a repeating animation or game with objects that repeatedly appear to interact with.**

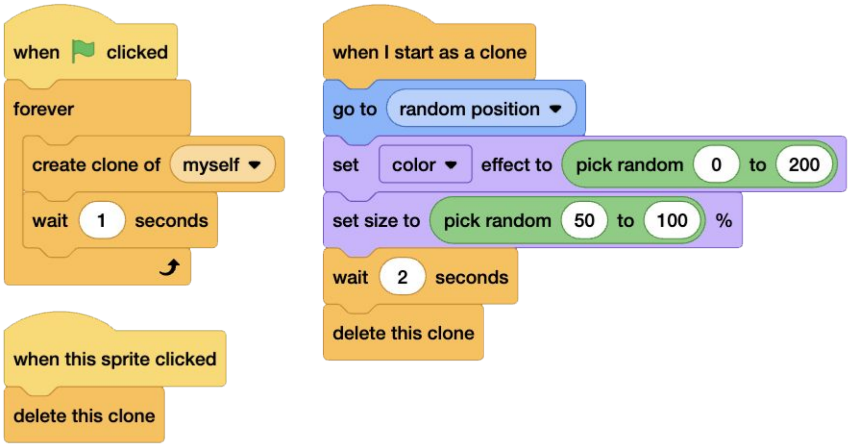
For instance, you could make an animation where balloon clones are created every few seconds and then disappear or can be popped by the user.

(Example here: scratch.mit.edu/projects/1154244503.)

Balloon Pop Clones

scratch.mit.edu

1. Create a script that makes a clone every few seconds forever when the green flag is clicked. You could…
    - have the original sprite move around and change color or size before creating a clone
    - have the clones adjust their own settings after they have been created

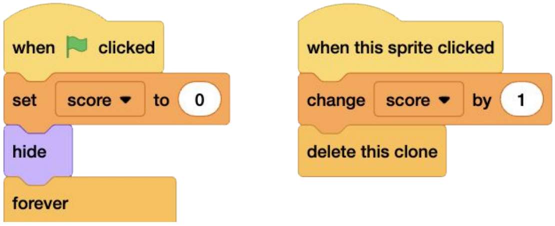


```
when 🏁 clicked
forever
    create clone of (myself ▾)
    wait (1) seconds
↩

when this sprite clicked
delete this clone

when I start as a clone
go to (random position ▾)
set [color ▾] effect to (pick random (0) to (200))
set size to (pick random (50) to (100)) %
wait (2) seconds
delete this clone
```

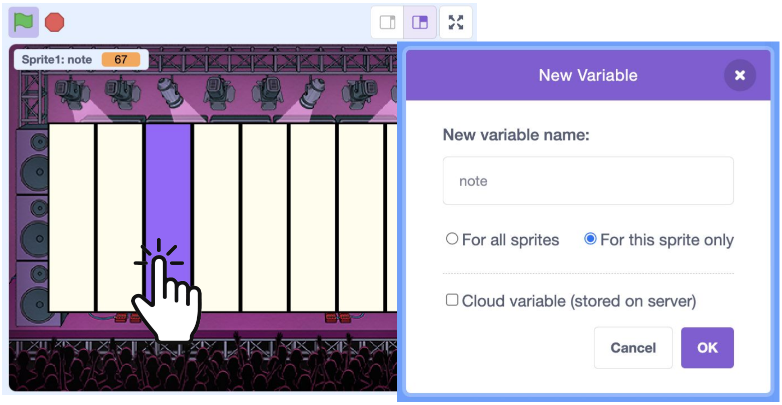
2. What if you want to add a score? Where would you add blocks to set and change the score? You may also need to hide the original and show the clones.



```
when 🏁 clicked
set [score ▾] to (0)
hide
forever

when this sprite clicked
change [score ▾] by (1)
delete this clone
```

Clone Piano: Using Local Variables



A local variable ("For this sprite only") is individual to a single sprite or a single clone. (Versus a global variable that applies to all sprites in the project and all their clones.)

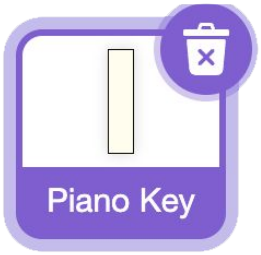
The value stored in each clone's local variable is the value that was present at the moment the clone was created.

Local variables show the sprite name followed by the variable name in the stage monitor.

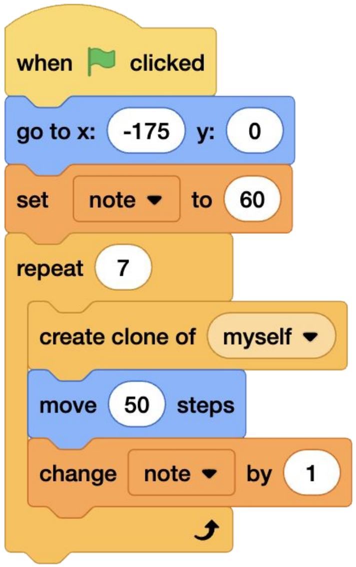
(Example here: scratch.mit.edu/projects/1181518635.)

Clone Piano: Using Local Variables

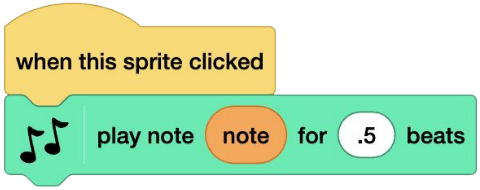
scratch.mit.edu



1. Create a piano key sprite by using the Rectangle drawing tool in the Paint Editor.
2. Assemble a script that creates clones of the piano key sprite in a row.
3. Create a local variable ("For this sprite only") to store the note for each clone and the original sprite.
4. Set the initial note, and then change the note after each clone is created. Use the "note" variable in the "play note" block. Test and debug!



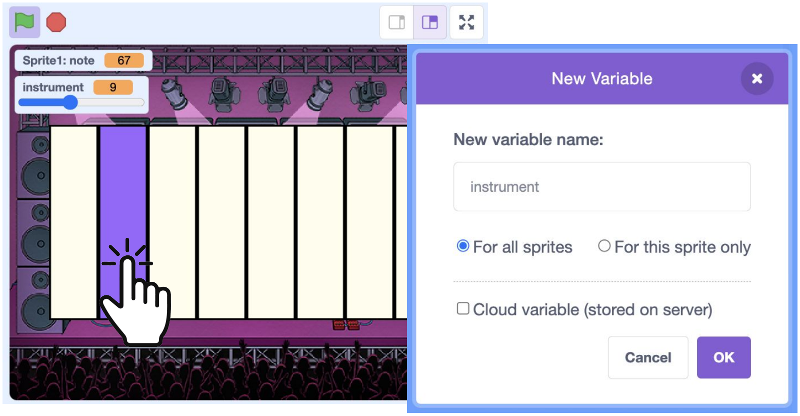
```
when [flag] clicked
go to x: (-175) y: (0)
set [note ▾] to (60)
repeat (7)
    create clone of (myself ▾)
    move (50) steps
    change [note ▾] by (1)
end
```



```
when this sprite clicked
play note (note) for (.5) beats
```

Optional: Adjust your program so it changes the color of the piano key or shows a different costume for the piano key as the note is played, so you can hear and see when a piano key is pressed.

Clone Piano: Using Global Variables



New variable name:

instrument

[✓ For all sprites] For this sprite only

Cloud variable (stored on server)

A global variable ("For all sprites") applies to all sprites in the project and all their clones. (Versus a local variable that is individual to a single sprite or a single clone.)

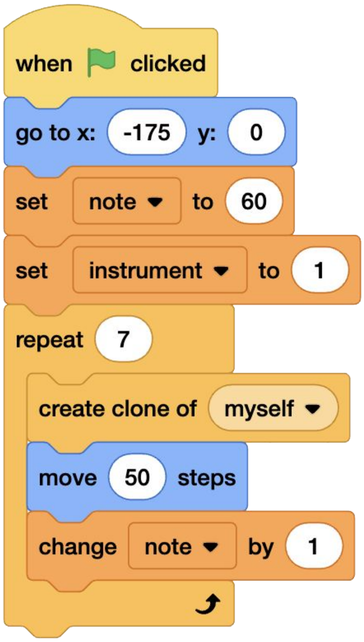
The value stored can be changed, and the variable is updated for all sprites in a project.

Global variables just show the variable name in the stage monitor.

(Example here: scratch.mit.edu/projects/1181518635.)

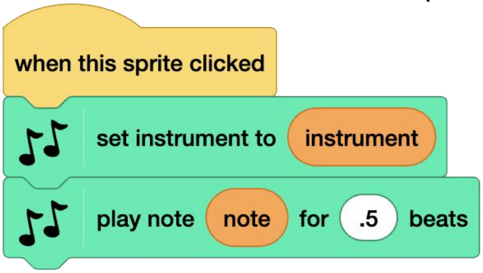
Clone Piano: Using Global Variables

scratch.mit.edu

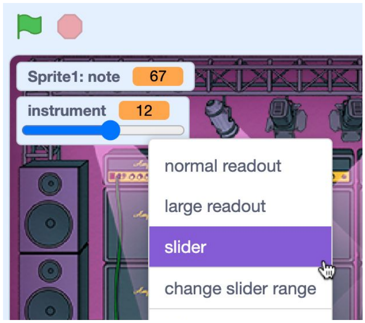


```
when green flag clicked
go to x: (-175) y: (0)
set [note] to (60)
set [instrument] to (1)
repeat (7)
    create clone of (myself)
    move (50) steps
    change [note] by (1)
end
```

1. Create a global variable (“For all sprites”) to store the instrument for each clone and the original sprite.
2. Set the initial instrument, and then use the “instrument” variable in the “set instrument” block.
3. Right click on the “instrument” stage monitor to change it to a slider. Then, right click again to set the range from 1-21 (the number of instruments available). Now, use the slider to change the instrument globally, for all piano keys. Test and debug!



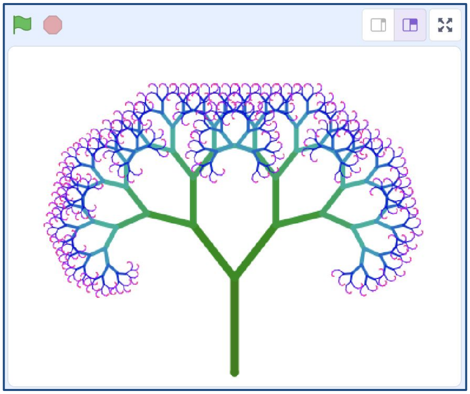
```
when this sprite clicked
set instrument to (instrument)
play note (note) for (.5) beats
```



Sprite1: note 67

instrument 12

Fractal Tree: Clones Making Clones



When a clone is created, the scripts for the original sprite also apply to a clone. That means that **clones can be coded to create clones**.

There is a limit to the number of clones a program can create. At this time, each program can create a maximum of 300 clones. The limit is important, because issues could arise with too many clones (for instance, it could cause the program to lag).

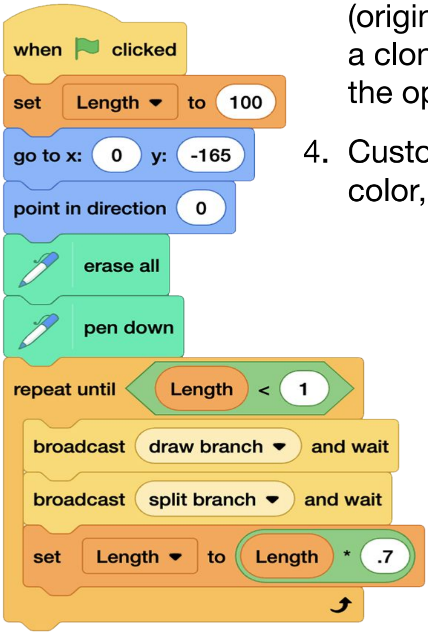
(Example here: scratch.mit.edu/projects/1145558837.)

Fractal Tree

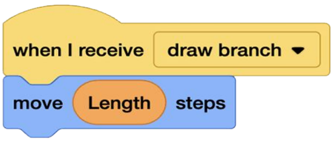
scratch.mit.edu

The fractal tree uses the pen tool and an army of clones to draw each branch.

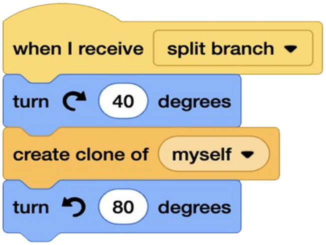
1. Set the initial length, position, and direction of the first branch (which will actually be the trunk of the tree). Then, place the pen down. (Pen blocks can be found in the Extensions.)
2. Next, repeatedly have the program draw a branch by moving, split the branch into two, and divide the length, so branches get shorter and more numerous.
3. To split the branch, have the sprite (original and all clones) turn, create a clone of itself, and then turn in the opposite direction.
4. Customize by changing the pen color, brightness, size, etc.



```
when 🏳 clicked
set Length ▾ to 100
go to x: 0 y: -165
point in direction 0
erase all
pen down
repeat until <(Length) < 1>
    broadcast draw branch ▾ and wait
    broadcast split branch ▾ and wait
    set Length ▾ to ((Length) * .7)
↺
```

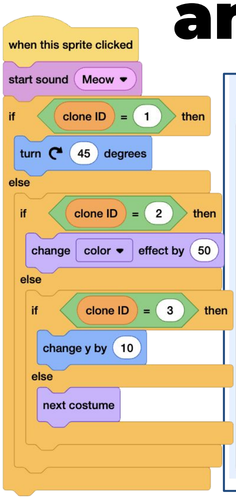


```
when I receive draw branch ▾
move (Length) steps
```

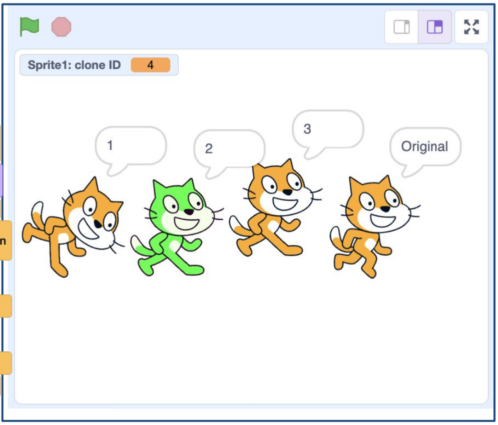


```
when I receive split branch ▾
turn ↻ 40 degrees
create clone of myself ▾
turn ↺ 80 degrees
```

Clone IDs: Generating and Using



```
when this sprite clicked
start sound (Meow ▾)
if <(clone ID) = (1)> then
    turn ↻ (45) degrees
else
    if <(clone ID) = (2)> then
        change (color ▾) effect by (50)
    else
        if <(clone ID) = (3)> then
            change y by (10)
        else
            next costume
```



Sprite1: clone ID 4

Do you want to have more control over an individual clone’s behavior? **Assign each clone an ID as it is created, then use that ID in a conditional statement to set unique code sequences for each clone.**

Generate clone IDs using a local variable (“For this sprite only”). The value stored in each clone’s local variable is the value that was present at the moment the clone was created.

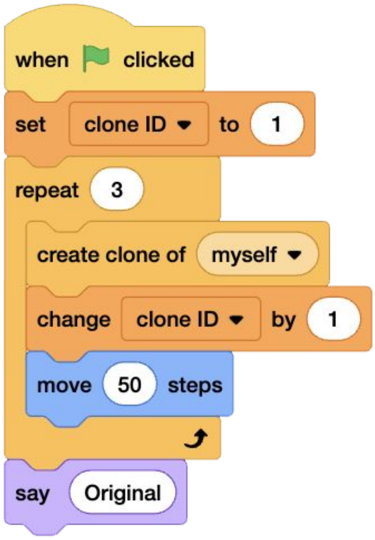
(Example here: scratch.mit.edu/projects/1184917851.)

Clone IDs

scratch.mit.edu

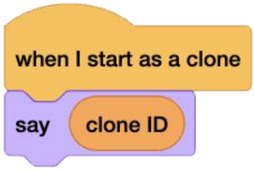
1. Create a local variable ("For this sprite only") to store the clone ID.

2. Set the clone ID to 1, then change the clone ID by 1 after each clone is created.



```
when [flag] clicked
set clone ID to 1
repeat 3
    create clone of myself
    change clone ID by 1
    move 50 steps
say Original
```

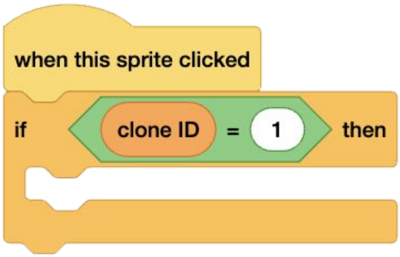
Optional: Have each clone say its clone ID, so you can easily identify them on the stage, You can always remove this code later, when finalizing the project.



```
when I start as a clone
say clone ID
```

3. Create a conditional statement and use the "equals" Operator block to identify individual clones by their clone ID.

   For example, use a series of "if then" blocks or nested "if then else" blocks with different scripts under each one.



```
when this sprite clicked
if clone ID = 1 then
```